Efficient parsing strategies for syntactic analysis of closed captions

Krzysztof Czuba
kczuba@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract

We present an efficient multi-level chart parser that was designed for syntactic analysis of closed captions (subtitles) in a real-time Machine Translation (MT) system. In order to achieve high parsing speed, we divided an existing English grammar into multiple levels. The parser proceeds in stages. At each stage, rules corresponding to only one level are used. A constituent pruning step is added between levels to insure that constituents not likely to be part of the final parse are removed. This results in a significant parse time and ambiguity reduction. Since the domain is unrestricted, out-of-coverage sentences are to be expected and the parser might not produce a single analysis spanning the whole input. Despite the incomplete parsing strategy and the radical pruning, the initial evaluation results show that the loss of parsing accuracy is acceptable. The parsing time favorable compares with a Tomita parser and a chart parser parsing time when run on the same grammar and lexicon.

1 Introduction

In this paper we present on-going research on parsing closed captions (subtitles) from a news broadcast. The research has been conducted as part of an effort to build a prototype of a real-time Machine Translation (MT) system translating news captions from English into Cantonese (Nyberg and Mitamura, 1997). We describe an efficient multi-level chart parser that was designed to handle the kind of language used in our domain within a time that allows for a real-time automatic translation. In order to achieve high parsing speed, we divided an existing English grammar into multiple levels. The parser proceeds in stages. At each stage, rules corresponding to only one level are used. A constituent pruning step is added between levels to insure that constituents not likely to be part of the final parse are removed. This results in a significant parse time and ambiguity reduction. Since the domain is unrestricted, out-of-coverage sentences are to be expected and the parser might not produce a single analysis spanning the whole input. Thus, the set of final constituents has to be extracted from the chart. Despite the incomplete parsing strategy and the radical pruning, the initial evaluation results show that the loss of parsing accuracy is acceptable. The parsing time favorable compares with a Tomita parser and a chart parser parsing time when run on the same grammar and lexicon.

The outline of the paper is as follows. In Section 2 we describe the syntactic and semantic characteristics of the input domain. Section 3 provides a short summary of previous published research. Section 4 gives an overview of requirements on the parsing algorithm posed by our application. Section 5 describes how the grammar was partitioned into levels. Section 6 describes the constituent pruning algorithm that we used. In Section 7 we present the method for extracting final constituents from the chart. Section 8 presents the results of an initial evaluation. Finally, we close with future research in Section 9.

2 Captioning domain

Translation of closed captions has been attempted before. (Popowich et al., 1997) describe a system that translates closed captions taken from North American prime time TV. In their approach, (Popowich et al., 1997) assume a shallow parsing method that proves effective in achieving broad system coverage required for translation of closed captions from, e.g., movies. As reported by (Popowich et al., 1997), the shallow analysis performed by the system combined with the transfer-based translation strategy results in a number of sentences that are understandable only in conjunction with the broadcast images. The number of sentences that are translated incorrectly is also significant.

The parsing scheme described below was used in a pilot Machine Translation system for translation of news captions. The following requirements were posed: a) the translations should not be misleading, b) they can be telegraphic since the input is often in a telegraphic style, c) partial translations are acceptable, d) if no correct translation can be produced then it is preferable to not output any.

The closed captions were taken from a financial news segment. Although the language in this segment is not strongly domain-restricted, it is centered around the financial aspects of the described events, which makes certain tasks such as sense disambiguation feasible.

In order to address the translation quality problems found by (Popowich et al., 1997), (Nyberg and Mitamura, 1997) propose a multi-engine MT system architechture to handle the task of translating closed captions. The parser described in this paper, was developed for the knowledge-based module (Nyberg and Mitamura, 1992) in the system and it was required to produce "deep" analyses with the level of detail sufficient for creating interlingua.

The stream of closed captions we worked with shows many interesting characteristics that influence the design of the parsing algorithm and the whole MT system. In the paper, we consider only the issues that are related to the syntactic analysis of closed captions. The stream contains long sentences taken from interviews, short phrases making the program more lively, and telegraphic style sentences used to report the latest stock market news. It has unrestricted syntax in long descriptive sentences, which resembles written language, with some phenomena like omission of function words that are characteristic of spoken language. It is likely to contain words not present in the lexicon, such as company names. Although not considered here directly, a caption stream usually also contains some typos and corrections made by the captioner. It is however different from, e.g., a speech recognizer output since the human captioner usually provides the correct transcription and there is no immediate need to model recognition errors.

3 Partial, robust and fast parsing

The kind of input described above requires robust parsing methods. Since our goal was real-time translation, the parsing module had to be very efficient. We concentrated on reducing the amount of work done by the parser at the potential cost of lowering the quality of the resulting analysis[1]. Similar methods have been adopted elsewhere, although in most cases the goal was a shallow parse. (Abney, 1996) describes a chunking approach based on cascades of finite transducers in which the parser finds "islands of certainty" at multiple analysis levels. Only maximal constituents (in terms of length) found by transducers at lower levels are the input to the transducers at higher levels, which results in constituent pruning and ambiguity containment, and low parsing times.

(Ciravegna and Lavelli, 1997) introduce additional controlling strategies to a chart parser. (Ciravegna and Lavelli, 1997) experiment with chunks and add a preprocessing step that uses a finite-state machine to identify potential chunks in the input. The chart parser control strategy is augmented so that constituents found during parsing that align with chunk boundaries are processed with a higher priority in the hope that they are more likely to become part of the final parse. Constituents that do not align with chunk boundaries are assigned lower priorities and remain in the agenda. The resulting algorithm can avoid pruning of useful constituents that can happen in Abney's cascading approach.

Explicit pruning of constituents is another strategy that can be used to improve the efficiency of a parser as shown, e.g., by (Rayner and Carter, 1996). (Rayner and Carter, 1996) experimented with a bottom-up parser with three levels: lexical level, phrasal level and full parsing level. Constituents are pruned before the phrasal and full parsing levels using statistical scores based on how likely is a constituent with a certain property to be part of the final solution.

4 Requirements on the algorithm

Given the characteristics of the input and previously published results described above, we decided to design a parsing strategy with the following characteristics:

1. bottom-up: this allows for partial parses to be identified;
2. multi-level: combined with pruning provides additional control over created constituents;
3. pruning: constituents not likely to contribute to the final parse should be removed early.

In what follows we will describe some initial results of experiments in applying a multi-level chart parser with a radical pruning strategy to the captioning domain.

5 Introducing levels to the grammar

The parsing proceeds in multiple levels. The grammar is split into groups of rules corresponding to the different levels. At a given level, only rules assigned to this level are allowed to fire. They use as input all the constituents the parser created so far (only inactive edges, all active edges are currently pruned between levels). In a sense, this corresponds to parsing with multiple grammars, although the grammar partitioning can be done once the grammar is complete and fine tuning is performed to achieve the right balance between ambiguity, output fragmentation and parsing speed. This approach makes it also possible to test the improvement introduced by the special processing scheme in comparison to some

[1]Dealing with typos, out-of-coverage lexical item, etc, was not considered here, although the parser offers some robustness in skipping unknown words, see Section 7.

other parsing scheme since the grammar is kept in a general form.

The grammar that we are currently using has been adapted from a general English grammar written for the Tomita parser. The grammar does not follow any particular theoretical framework, although it has been strongly influenced by both LFG (Bresnan, 1982) and HPSG (Polard and Sag, 1994). It consists currently of about 300 general rules with some general attachment preference constraints that have been fine tuned on various kinds of text, including news broadcasts and written texts. One important characteristic of the grammar is the use of subcategorization information (at the syntactic level, with values such as `subject+object`, `subject+complement`, `subject+object+oblique("of")`, etc.), and some broad semantic classification of adjuncts to help prevent ambiguity. The lexicalist character of the grammar requires that subcategorization and simple semantic information (temporal expressions, location expressions, etc) be present in the lexicon. The grammar coverage (Czuba et al., 1998) is quite broad, but it is not sufficient, e.g., to cover some types of clause attachment that are present in long sentences found in the input.

The changes that were introduced to the grammar, in comparison to the original version without levels, concentrated on a simple addition of levels to the rules and duplicating some rules at multiple levels. In the current grammar, the following levels have been introduced:

1. lexical level: lexicon lookup, idioms and fixed phrases;
2. nominal phrases without adjuncts;
3. verb phrases with obligatory complements;
4. noun and verb phrases with adjuncts;
5. clausal constituents; noun and verb phrases with obligatory clausal complements;
6. top level with preferences for attachment and well-formedness of constituents (e.g., prefer finite clauses, prefer ungapped constituents, etc).

Although some motivation for the above partitioning can be found on X-bar theory, we mostly used our intuition for choosing the number of levels and deciding how to assign rules to different levels. These are parameters of the algorithm and they can be tuned in further experiments. See the next section for examples of rules that were added to multiple levels.

6 Constituent pruning

We will refer to constituents that were created using grammar rules with the nonterminal `<X>` on their LHS as `<X>` *constituents*. Two constituents will be *similar* if they were created using rules with the same nonterminal on the LHS. E.g., two `<NP>` constituents are similar, regardless of their positions and spanned input.

The pruning phase was added to the usual chart parsing algorithm in a way that makes it invisible to the grammar writer. The pruning algorithm is called on the chart whenever no more rules are allowed to fire at the current level and no active arcs can be extended. In the initial implementation, the pruning algorithm was based on a simple subsumption relation: only the maximal(i.e., covering the longest number of input tokens) constituents from a set of similar constituents remain in the chart and are added to the agenda for the next level. E.g., if the chart contained two `<NP>` constituents, only the one spanning more input would be retained.

Although the original pruning strategy resulted in many reasonable parses, we noticed a few general problems. The parser is very sensitive to wrongly created long constituents. This means that the grammar has to be relatively tight for a successful application with the described parser since no global optimization is performed. However, this also means that if in a particular context the parser builds a constituent C that is not correct in this context but the rules that were used to build the constituent cannot be removed from the grammar since they are useful in general, the pruning step will wrongly remove all, potentially correct, similar constituents subsumed by C.

We observed this kind of behavior in practice and at least two cases can be distinguished. Some constituents are added to the chart early in the analysis and they form bigger constituents as the analysis progresses. Consequently, if a similar constituent is created, the original constituent will be pruned and will not be available at higher levels. This behavior can be observed, e.g., for the string *that help*, in which *that* is supposed to be a pronoun and *help* is supposed to have a verbal reading. Since *that help* is a well-formed `<NP>` constituent according to our grammar, it will be created and it will subsume the pronoun *that*. This means that the pronoun *that* will be pruned and it will not be available at a higher level that tries to create a correct parse incorporating *that* as an object of a preceding verb and *help* as the main verb. In order to solve this problem we allowed some rules to be repeated at multiple levels. The rules introducing pronouns were added at two levels. The rules involving verbal complements were also introduced twice. Since verbal phrases are created relatively late in the analysis, verbal complements on, e.g., noun phrases are not available yet. Because of that, e.g., the rule that attaches the direct object to a verb is present twice in the grammar:

one version of it takes care of nominal objects without complements, the other one is specific for objects with a verbal complements. Since the second version of the rule contains a check for the presence of a verbal complement, no work is repeated.

The second case that we noticed involved large noun phrases created as the result of applying the rules for nominal apposition (e.g., *the U.S. president, Bill Clinton*) and coordination. Since the parser does not use any semantic information, it is difficult to prevent such rules from applying in some wrong contexts. Examples include noun phrases at clause boundaries as in the following sentence: *BTM says it will issue new shares to strengthen its capital base, BTM plans to raise 300 billion yen via the issue.* In this case, an apposition *its capital base, BTM* is created and the phrases *its capital base* and *BTM* are pruned, preventing the parser from finding the correct analysis consisting of two finite clauses.

In order to solve that problem, we introduced an option of relaxing the pruning constraint. Currently, such relaxing is allowed only for phrases containing appositions and coordination. All constituents that are subsumed by similar ones containing apposition or coordination are marked and they can never be pruned. As a result, both *its capital base* and *BTM* remain in the chart and can be used to create the required clauses at higher levels.

The pruning algorithm that we implemented can be potentially quite costly since it involves many comparisons between constituents. Although its worst-case cost is quadratic, in practice the equivalence classes of similar constituents are small and they are pruned quickly. As a result, in our experiments the pruning time was below 0.01% of the total parse time.

In addition to the actual removal of constituent, the function implementing the pruning algorithm performed local ambiguity packing: it removes constituents that have the same feature structures as a constituent already present in the chart and it creates constituents with disjunction of feature structures in case similar constituents spanning the same chunk of input but having different feature structures are found.

7 Extraction from the chart

After the parser finished creating constituents at the highest level, the final result has to be extracted from the chart. Since the parser might not be able to produce a single analysis spanning the whole input, the best sequence of constituents needs to be extracted.

Currently, a simple best-first beam search through the chart is used to find a sequence (path) of constituents spanning the whole input. Paths are allowed to have gaps, i.e., they do not have to be contiguous, although we do not allow for overlapping constituents. The algorithm prefers shorter paths. The length of a path is computed as a weighted sum of the lengths of constituents in the path. We experimented with two different ways of assigning weights and lengths to constituents. In the first method, each constituent was assigned the length of 1 that was weighted by a factor depending of the "quality" of the constituent. Paths can be extended by a gap spanning one input token at a time. Such a gap is weighted with the factor of 3. Constituents that are created by rules with the nonterminal `<OUTPUT>` on their LHS are assumed to be of the highest quality and they are weighted with the factor of 1. All remaining constituents can also be added to the path and are weighted with the factor of 1.5. So a path consisting of an `<OUTPUT>` constituent spanning input tokens 1 to 3, a gap spanning input tokens 4 and 5, and a `<V1>` constituent spanning input tokens 6 to 10 would receive the length of $1 + 6 + 1.5 = 8.5$. This algorithm shows a strong preference for contiguous paths and assigns lengths depending on the number of constituents in the path, ignoring their length.

The second weighting scheme we tried was based on the actual length of constituents. `<OUTPUT>` constituents were assigned their actual length multiplied by 1. Other constituents had their actual length multiplied by 1.5, and gaps were weighted with the factor of 2. The path described in the previous paragraph was thus assigned the length of $3 + 4 + 7.5 = 14.5$.

Although the first weighting scheme seems rather crude, it resulted in a very good performance both in terms of the quality of paths found and the time required to find the best path. The best-first search was implemented using a binary heap priority queue in an array, and the extraction time for the first weighting scheme was below 5% of the total time required for both parsing and extraction. We also did not notice any cases in which the returned path was not the optimal one given the constituents found by the parser. The second weighting scheme is more fine-grained and might turn out to be better on a bigger corpus. However, it required about 15 as much time to complete the search as the first scheme.

Finally, in case of ambiguity, the first feature structure returned by the parser was chosen.

8 Preliminary results

The algorithm has been applied to a sample of 42 sentences from a real TV broadcast. The sentences were picked from a contiguous transcript that was cleaned up for captioner mistakes. Since the parser was designed for use in a real-time multi-engine MT system, we concentrated on sentences which were likely to be translated by the knowledge-based part of the system. Sentences that were likely to be

Sentence 1, 17 tokens:
[[Well] [[Japan] [is [unlikely to adopt [any more stimulus spending measures]] soon [despite [that U.S. pressure.]]]]]

algorithm	#arcs	#constituents	time (s)
chart	20410	6680	76.6
Tomita	-	-	28.0
multi-level	891	191	1.0

Sentence 2, 49 tokens:
[[Reform legislation] [is quite good]] [because [it [[puts up] [public money]] [which] [[financial institutions] [can get]] [to protect [depositors]] [but only if], [[[they] [[[recycle] [or if you will,] [write off]] [their bad loans]]] [and] [[clean up] [their balance sheets]]] [[so] [they] [can start [to loan [money]] again.]]

algorithm	#arcs	#constituents	time (s)
chart	13530	4849	43.7
Tomita	–	–	22.0
multi-level	3101	388	3.0

Figure 1: Sample sentences with bracketing found the parser

translated by a translation memory look-up, such as greetings and short canned phrases, were not added to the test corpus. The resulting corpus consisted of relatively long sentences, with the average length of 23.5 tokens (including punctuation).

The parser was compared to a Tomita parser and a chart parser with local ambiguity packing. All the parsers were implemented in lisp and used the same unification package, which made the parsing results easily comparable. They also used the same grammar and lexicon with about 8000 entries. The grammar was preprocessed for the Tomita parser and the chart parser by removing the rules that were present at multiple levels. The chart parser was set up to quit after finding the first parse covering the whole input. All times given below were obtained on the same machine (SPARCStation 5). Care was taken to disable lisp garbage collection during parsing.

As it was expected, the pruning strategy resulted in a significant reduction of execution time. In Figure 1 we present a few measurements to illustrate the time improvement for two example sentences. For the first one, all parsers produced a complete analysis spanning the whole input. For the second one, due to the lack of grammar coverage, no single analysis can be found. Figure 1 shows the bracketing that the multi-level parser found. It also produced correct feature structures for all the chunks that can be used by an MT system.

As can be seen from Table 1, the time improvement is significant. The improvement in the number of elements in the final chart is crucial for good performance of the extraction algorithm that chooses constituents to be output by the parser in case there is no single analysis spanning the whole input. Also, the multilevel parser did not produce ambiguous feature-structures (ambiguity containment). The Tomita parser produced two packed f-structures for the first sentence that would have to be unpacked and disambiguated for further processing. For the second sentence, the Tomita parser did not find any full parse. The multilevel parser produced chunks that are usable in an MT system and can be translated giving at least partial translation to the user.

The results on the whole test set were as follows. The Tomita parser needed 652 seconds to analyze all the sentences. It produced a complete analysis for 31 sentences, returning no analysis for 11. The chart parser run till the first solution spanning the whole input was found needed 937 seconds to analyze the same 31 sentences, failing on the rest. In the case of the Tomita parser, the average ambiguity level was 3.7 analyses per sentence. The Tomita parser produced an acceptable[2] parse for all the 31 sentences it could analyze in full. However, the acceptable analysis would still have to be distilled from all the parses it returned. The time needed by the chart parser was prohibitively high for any attempt at extracting constituents in the cases when no single parse was found. The total time required by the multi-level parser was 60.7 sec, 10.7 times less than the Tomita parser and 15.4 times less than the chart parser. Figure 2 illustrates the parsing and extraction time as function of sentence length. Although clearly dependent on the syntactic complexity of the input sentence, the parsing time appears to be linearly related to the input length.

[2]An analysis with, e.g., wrong PP-attachment that could be potentially repaired using lexical selection rules during translation, was marked as acceptable.

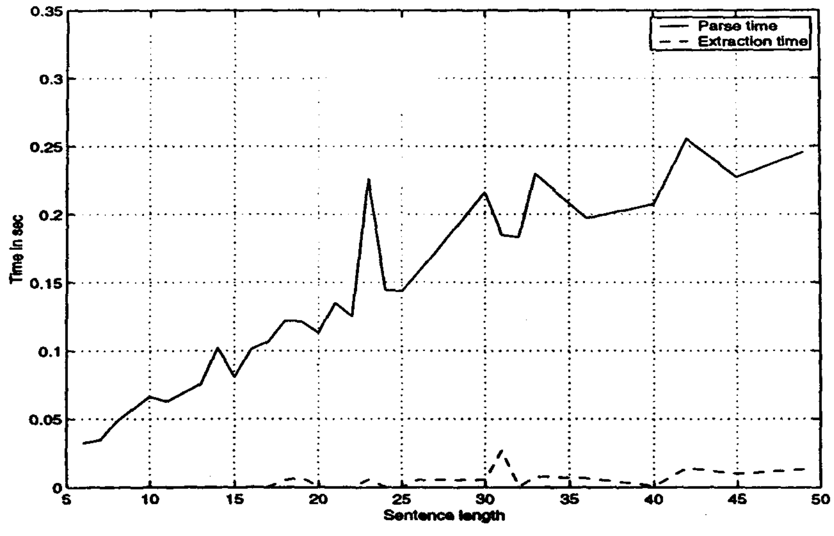

Figure 2: Parse and extraction time as function of sentence length

Since the parser's ability to attach constituents is limited, it produced 114 chunks for all the input sentences, average 2.71 per sentence. The chunks fully covered the input sentences. They were compared to a bracketing that was assigned by a human and corresponded to linguistically motivated phrases. Out of these, 11 corresponded to a wrong bracketing as judged by a human. This affected 8 sentences in the corpus. The remaining chunks were acceptable (as defined in the footnote on the previous page). Although an evaluation in terms of precision/recall would be possible, we have not done it for this paper. We believe that an end-to-end evaluation using, e.g., an MT or Information Extraction system that would be able to handle the parser output would be more meaningful, since it would also be a way to evaluate the effect of the large number of small chunks the parser produced.

9 Future research

There are a number of research issues that we are planning to address soon. First, a more thorough evaluation is required, as described in the previous section. We are currently looking for ways to perform such an evaluation. We are also looking into replacing the fixed pruning and constituent extraction strategy with one learned from training data. We are also investigating ways of learning the number of levels and grammar rule partitioning among levels.

10 Acknowledgements

We would like to thank Eric Nyberg and Teruko Mitamura for providing support for this research and useful discussion, Alon Lavie for helpful comments about the research and earlier versions of the paper, and the anonymous NAACL Student Workshop reviewers for their constructive comments.

References

S. Abney. 1991. Parsing by chunks. In Berwick, Abney, and Tenny, editors, *Principle-Based Parsing*. Kluwer.

S. Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of Workshop on Robust Parsing, ESSLI-96*.

J. Bresnan, editor. 1982. *The mental representation of grammatical relations*. MIT Press.

F. Ciravegna and A. Lavelli. 1997. Controlling bottom-up chart parser through text chunking. In *Proceedings of IWPT'97*.

K. Czuba, T. Mitamura, and E. Nyberg. 1998. Can practical interlinguas be used for difficult analysis problems? In *Proceedings of AMTA-98 Workshop on Interlinguas*.

E. Nyberg and T. Mitamura. 1992. The kant system: Fast, accurate, high-quality translation in practical domains. In *Proceedings of COLING-92*.

E. Nyberg and T. Mitamura. 1997. A real-time MT system for translating broadcast captions. In *Proceedings of MT Summit VI*.

C. Polard and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

F. Popowich, D. Turcato, O. Laurens, and P. McFetridge. 1997. A lexicalist approach to the translation of colloquial text. In *Proceedings of TMI-97*.

M. Rayner and D. Carter. 1996. Fast parsing using pruning and grammar specialization. In *Proceedings of ACL'96*.